



# MongoDB 기초

## 02 CRUD



## 목차

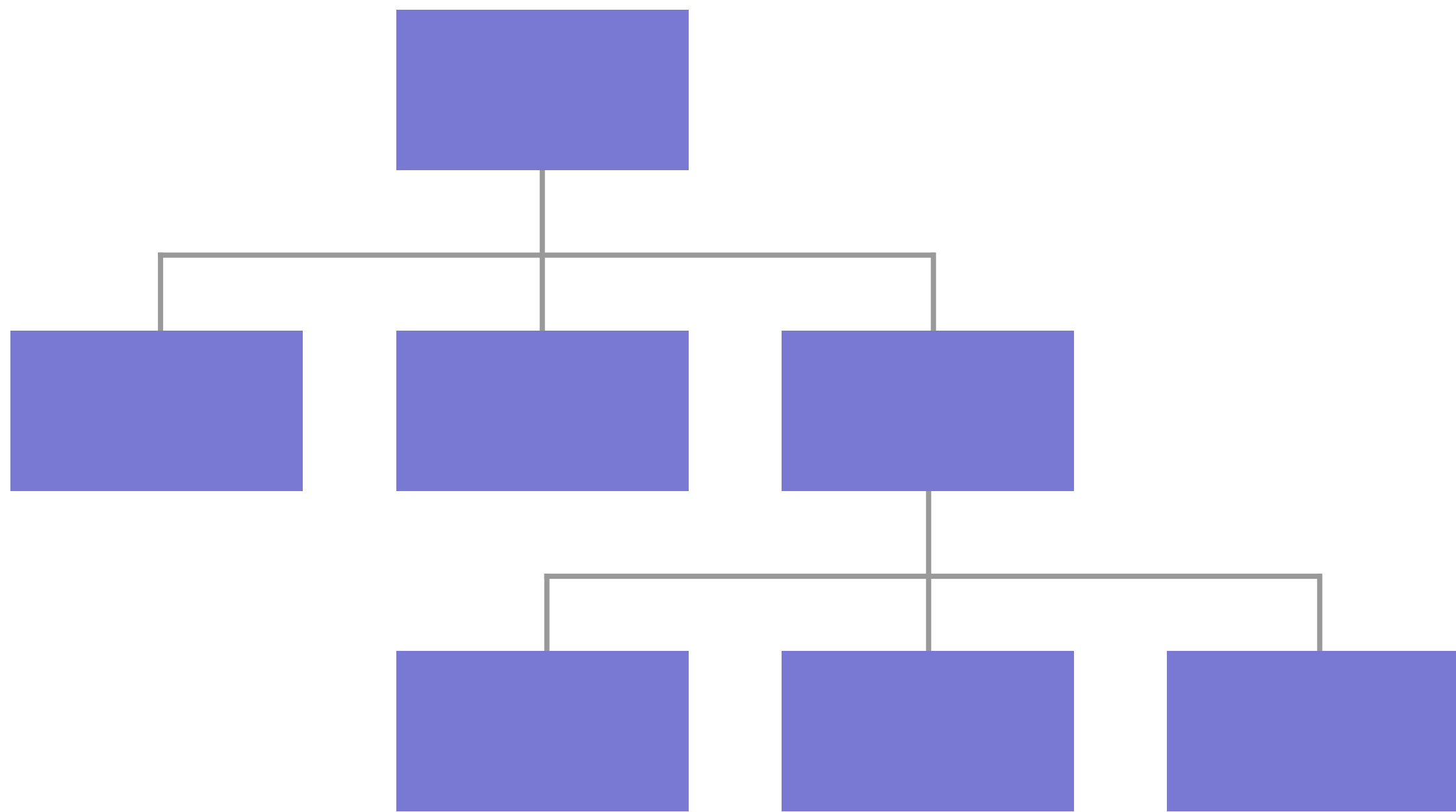
- 01. MongoDB의 구조
- 02. BSON 데이터 타입
- 03. 도큐먼트 생성
- 04. 도큐먼트 검색 기초
- 05. 도큐먼트 수정
- 06. 도큐먼트 삭제

01

# MongoDB의 구조



✓ MongoDB의 기본 구조



데이터베이스

컬렉션

도큐먼트

## ✓ JSON과 비슷한 BSON 자료구조

### BSON 구조 예시

```
{  
  name: "sue",  
  age: 26,  
  status: "A",  
  groups: ["news", "sports"]  
}
```

name: "sue",	←	field: value
age: 26,	←	field: value
status: "A",	←	field: value
groups: ["news", "sports"]	←	field: value

JSON과 유사한 BSON 구조로 정보를 저장

Javascript Object Notation

## ✓ Pymongo 소개

### Python 코드

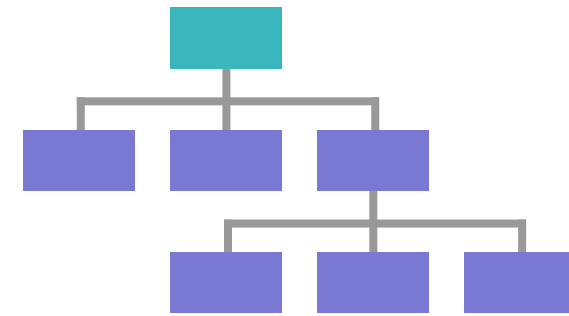
```
import pymongo

connection = pymongo.MongoClient("mongodb://localhost:27017/")
```

내 PC 내부      MongoDB  
기본 포트

pymongo는 mongoDB를 사용할 수 있게 도와주는 파이썬 모듈

## ✓ Pymongo로 DB 접속



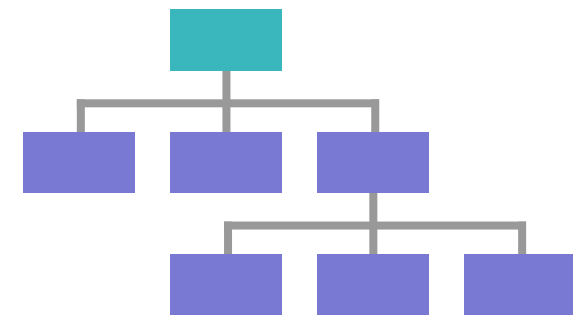
## Python 코드

```
import pymongo
connection = pymongo.MongoClient("mongodb://localhost:27017/")

db = connection.get_database("testDB")
```

접속할 데이터베이스로 접근  
만약 데이터베이스가 없으면 자동으로 생성한 후 접속

## ✓ 컬렉션에 문서 삽입하기



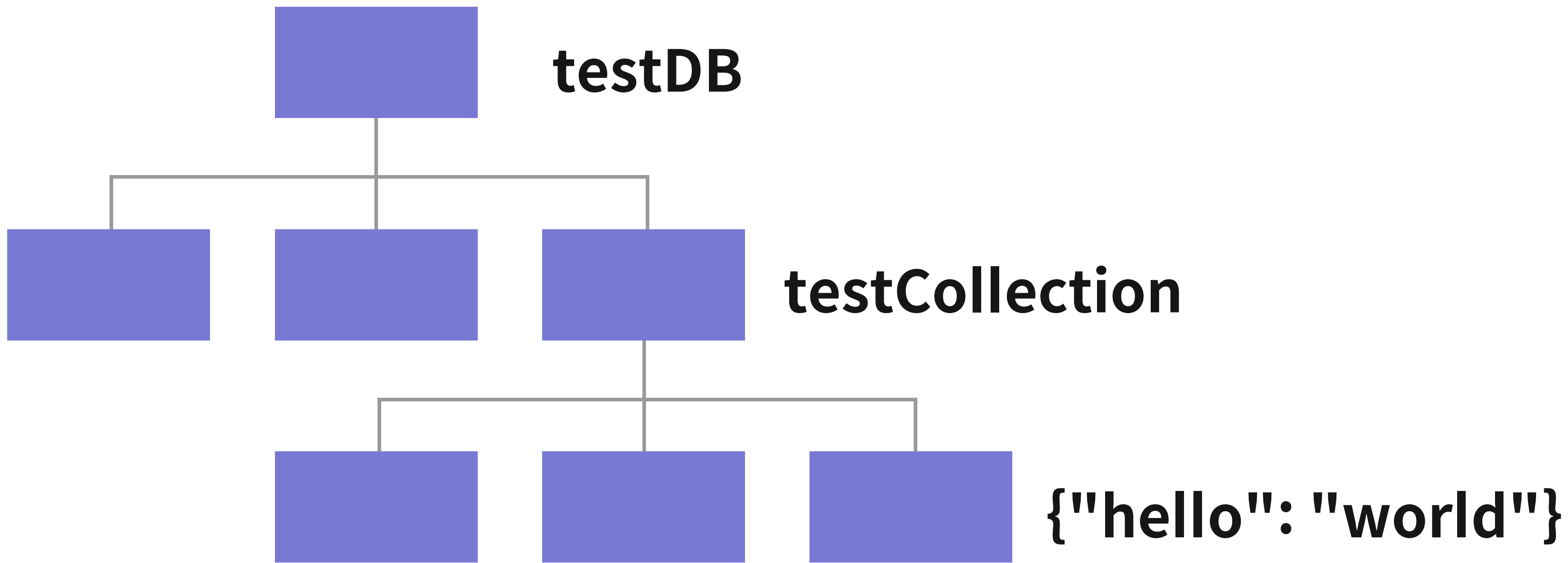
## Python 코드

```
import pymongo
connection = pymongo.MongoClient("mongodb://localhost:27017/")
db = connection.get_database("testDB")
collection = db.get_collection("testCollection")
collection.insert_one({ "hello": "world" })
```

컬렉션에 문서 저장  
만약 컬렉션이 없다면 자동으로 생성됨



✔ MongoDB의 기본 구조



## ✓ 데이터베이스, 컬렉션, 도큐먼트 확인하기

### Python 코드

```
# 데이터베이스 목록 조회
print(connection.list_database_names())

# 컬렉션 목록 조회
print(db.list_collection_names())

# pprint로 도큐먼트 목록 조회
pprint(list(collection.find()))
```

Cursor 출력

### 결과 출력

```
['admin', 'config', 'local', 'testDB']
```

기본 DB들

```
['testCollection']
```

```
[{'_id': ObjectId('...'), 'hello': 'world'}]
```

도큐먼트를 식별하기 위한 값


02

# BSON 데이터 타입



## ✓ BSON이란?

```
{  
  name: "karoid",  
  age: 26,  
  status: "B",  
  groups: ["family", "sports"],  
}
```



field: value  
field: value  
field: value  
field: value

## BSON = Binary JSON의 의미

JavaScript object notation

JSON의 일부로써 MongoDB 문서로  
데이터를 저장하기 위한 형식

✔ BSON에 들어가는 자료형

Type	Number	Alias	Notes
Double	1	“double”	
String	2	“string”	
Object	3	“object”	
Array	4	“array”	
Binary data	5	“binData”	
Undefined	6	“undefined”	Deprecated.
ObjectId	7	“objectId”	
Boolean	8	“bool”	
Date	9	“date”	
Null	10	“null”	
Regular Expression	11	“regex”	

Type	Number	Alias	Notes
DBPointer	12	“dbPointer”	Deprecated.
JavaScript	13	“javascript”	
Symbol	14	“symbol”	Deprecated.
JavaScript (with scope)	15	“javascriptWith Scope”	
32-bit integer	16	“int”	
Timestamp	17	“timestamp”	
64-bit integer	18	“long”	
Decimal128	19	“decimal”	New in version 3.4.
Min key	-1	“minKey”	
Max key	127	“maxKey”	

✓ **BSON의 데이터타입**

**NULL**

**아무것도 없다**

**Undefined**

**정의 되지 않음**

## ✓ BSON의 데이터타입

**Double/Integer**

**123.42, 12**

**String**

**"hello" / 'hello'**

**Object**

**{field: "value"}**

**Array**

**[1,2,{hi:"hello"}]**

**Boolean**

**true / false**

## ✓ BSON의 데이터타입

**Date**

`ISODate("2017-10-24T05:02:46.395Z")`  
UTC Time

**ObjectId**

`ObjectId("542c2b97 bac059 5474 108b48")`



## ✓ ObjectId의 구성

# ObjectId

MongoDB에서 각 Document 의 primary key의 값으로 사용된다

ObjectId( "542c2b97 bac059 5474 108b48" )

유닉스 시간      기기 id      프로세스 id      카운터

## ✓ 정리하기

### BSON 구조 예시

```
{
  _id: ObjectId("542c2b97 bac059 5474 108b48"),
  name: {first: "sue", last: "Turing" },
  age: 26,
  is_alive: true,
  groups: ["news", "sports"],
  viewTime: ISODate("2017-10-24T05:02:46.395Z")
}
```

## ✓ Pymongo에서 사용하기

Python으로 표현한 자료형

```
from bson import ObjectId      ← bson에만 있는 자료형
from datetime import datetime ← python 내부 자료형
collection.insert_one({
    "_id": ObjectId("542c2b97bac0595474108b48"),
    "name": {"first": "sue", "last": "Turing" },
    "age": 26,
    "is_alive": True,
    "groups": ["news", "sports"],
    "viewTime": datetime(2017, 10, 24, 5, 2, 46)
})
```

03

# 도큐먼트 생성



## ✓ 문서를 보기 좋게 출력하기

Python 코드

```
from pprint import pprint  
pprint({ BSON document })
```

pretty print의 의미를 가진 명령어 pprint

## ✓ 컬렉션에 문서 삽입하기

### Python 코드

```
import pymongo
connection = pymongo.MongoClient("mongodb://localhost:27017/")
db = connection["testDB"]
collection = db["testCollection"]
collection.insert_one({ "hello": "world" })
```

이전에 잠깐 배웠을 때  
insert\_one으로 문서를 생성했었다

## ✓ 컬렉션에 하나의 문서 삽입하기

### 명령어

```
from pprint import pprint
result = collection.insert_one(
    { document }
)
print(result.inserted_id)
pprint(result.inserted_id)
```

### 결과

```
# 입력된 문서의 _id 값
609fce475cfb9675580a6efc
ObjectId('609fce475cfb9675580a6efc')
```

하나의 문서 객체를 넘긴다

## ✓ 컬렉션에 다수의 문서 삽입하기

### 명령어

```
result = collection.insert_many(  
    [ { document }, { document }, ... ]  
)  
print(result.inserted_ids)
```

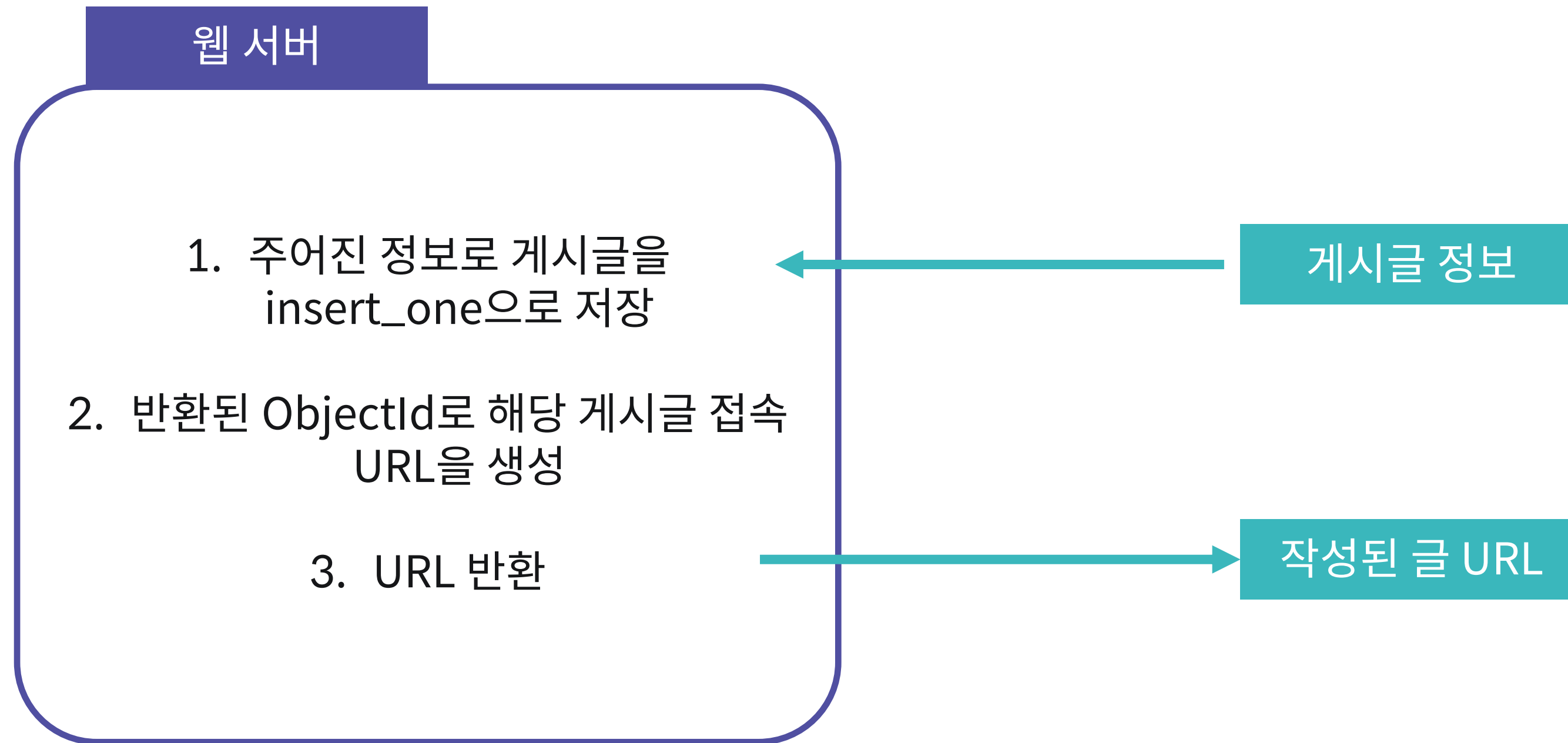
### 결과

```
# 입력된 문서의 _id 값들  
[  
    ObjectId('542c2b97bac0595474108b48'),  
    ObjectId('609fcdb30c1a70ffb15f4306')  
]
```

다수의 문서 객체를 넘긴다



## ✓ 웹 서버에서 문서 생성 예시



만약 게시글을 작성하는 웹 서버를 구현한다고 가정하면  
다음과 같은 로직을 만들 수 있다

04

# 도큐먼트 검색 기초



## ✓ 컬렉션에서 문서 조회

### Python 코드

```
import pymongo
from pprint import pprint
connection = pymongo.MongoClient("mongodb://localhost:27017/")
db = connection["testDB"]
collection = db["testCollection"]
collection.insert_one({ "hello": "world" })
print(list(collection.find()))
```

이전에 잠깐 배웠을 때  
find로 컬렉션 내의 문서를 조회했었다

## ✓ 컬렉션에서 문서 검색하기

### 명령어

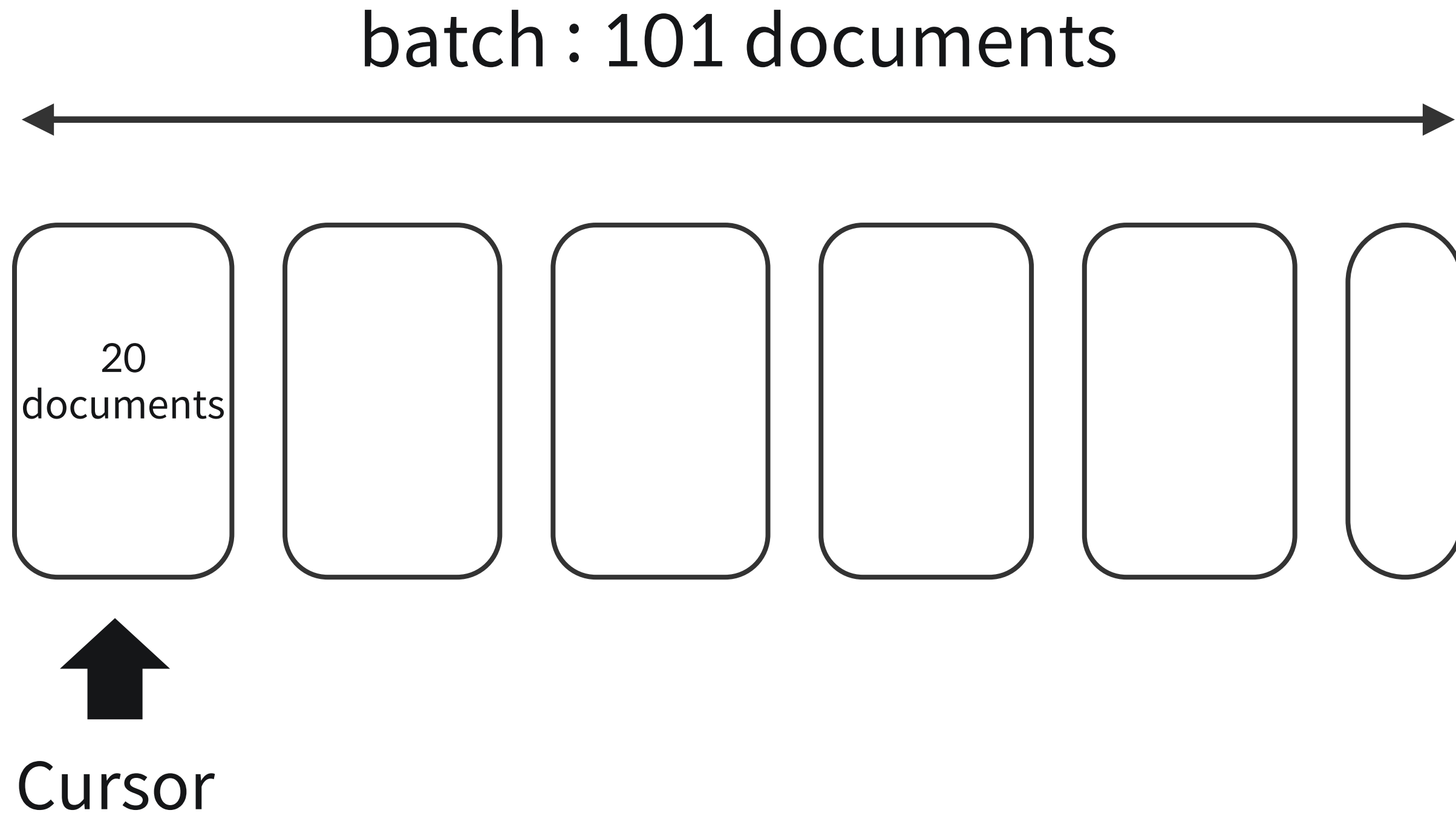
```
result = collection.find(  
    { query },  
    { projection }  
)  
print(result)  
print(list(result))
```

### 결과

```
# Cursor를 반환  
<pymongo.cursor.Cursor object at  
0x7fc6b31659b0>  
# list로 내용물을 불러올 수 있다  
[ { document }, { document }, ... ]
```

find 명령어는 컬렉션 내에 query 조건에 맞는  
다수의 문서를 검색한다

## ✓ 커서란?



커서란 **쿼리 결과에 대한 포인터**

문서 검색의 위치정보만을 반환하여 **작업을 효율적**으로 만들어준다

## ✓ 커서에서 문서 불러오는 부분

### 명령어

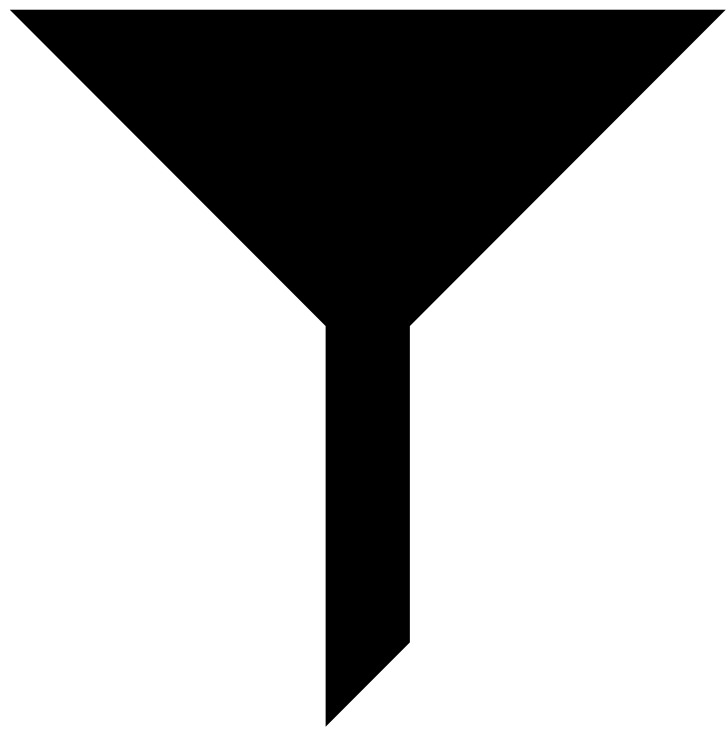
```
result = collection.find(  
    { query }  
)  
print(list(result))  
for document in result:  
    print(document)
```

### 결과

```
# list로 내용을 한번에 불러올 수 있다  
[ { document }, { document }, ... ]  
# for문으로 내용을 하나씩 처리할 수 있다  
{ document }  
{ document }  
...
```

list 명령어로 커서를 활용해 모든 데이터를 불러온다  
for 문으로 하나씩 데이터를 불러올 수 있다

## ✓ 쿼리란?



쿼리란 원하는 정보를 걸러내기 위한  
**칼때기**이다

검색하고자 하는 내용을  
쿼리로 표현할 수 있어야 한다

## ✓ 기본 Query

```
{"field": value, "field": value, ...}
```

**Query는 그 field에 맞는 value 값으로 필터링 한다!**



## ✓ find문 예시

```
users.find(  
  {"username": "karoid"}  
)
```

## 결과 Document

```
{ "_id" : ObjectId("59ef45b4ddf91a3b998ee9ed"), "username" : "karoid", "password" : "1111" }
```

## ✓ find문 예시

```
users.find(  
  {"password": "1111"}  
)
```

## 결과 Document

```
{ "_id" : ObjectId("59ef45b4ddf91a3b998ee9ed"), "username" : "karoid", "password" : "1111" }  
{ "_id" : ObjectId("59f02d5a36e39687dea2cea2"), "username" : "hello", "password" : "1111" }
```

## ✓ Projection이란?

```
{"field": boolean, "field": boolean, ...}
```

**Projection은 그 field를 보여줄지 말지를 알려준다**

boolean이 **true**이면 해당 **field**를 **표현**하고  
**false**면 **field**를 **제외**한 결과를 출력한다

## ✓ Projection 예시

```
users.find(  
  {"username": "karoid"},  
  {"username": True}  
)
```

## 결과 Document

```
{ "_id" : ObjectId("59ef45b4ddf91a3b998ee9ed"), "username" : "karoid" }
```

## ✓ Projection 예시

```
users.find(  
  {"password": "1111"},  
  {"username": False}  
)
```

## 결과 Document

```
{ "_id" : ObjectId("59ef45b4ddf91a3b998ee9ed"), "password" : "1111" }  
{ "_id" : ObjectId("59f02d5a36e39687dea2cea2"), "password" : "1111" }
```

## ✓ Projection의 잘못된 예

```
users.find(  
  {"password": "1111"},  
  {"username": False, "_id": True}  
)
```

## 결과 Document

ERROR!!

05

# 도큐먼트 수정



## ✓ 하나의 문서를 찾아 수정하기

### 명령어

```
result = collection.update_one(  
    { query },  
    { update },  
    upsert: Boolean  
)  
print(result.matched_count)  
print(result.modified_count)
```

### 결과

```
# 찾은 문서 수  
1  
# 변경된 문서 수  
1
```

**query로 검색하고, update에 변경할 사항을 적는다**



## ✓ 다수의 문서를 찾아 수정하기

### 명령어

```
result = collection.update_many(  
    { query },  
    { update },  
    upsert: Boolean  
)  
print(result.matched_count)  
print(result.modified_count)
```

### 결과

```
# 찾은 문서 수  
12  
# 변경된 문서 수  
5
```

**query로 검색하고, update에 변경할 사항을 적는다**

## ✓ 이미 이 문서가 들어있다고 가정하자

```
inventory.insert_many( [  
    { "item": "canvas", "qty": 100, "size": { "h": 28, "w": 35.5, "uom": "cm" }, "status": "A" },  
    { "item": "journal", "qty": 25, "size": { "h": 14, "w": 21, "uom": "cm" }, "status": "A" },  
    { "item": "mat", "qty": 85, "size": { "h": 27.9, "w": 35.5, "uom": "cm" }, "status": "A" },  
    { "item": "mousepad", "qty": 25, "size": { "h": 19, "w": 22.85, "uom": "cm" }, "status": "P" },  
    { "item": "notebook", "qty": 50, "size": { "h": 8.5, "w": 11, "uom": "in" }, "status": "P" },  
    { "item": "paper", "qty": 100, "size": { "h": 8.5, "w": 11, "uom": "in" }, "status": "D" },  
    { "item": "planner", "qty": 75, "size": { "h": 22.85, "w": 30, "uom": "cm" }, "status": "D" },  
    { "item": "postcard", "qty": 45, "size": { "h": 10, "w": 15.25, "uom": "cm" }, "status": "A" },  
    { "item": "sketchbook", "qty": 80, "size": { "h": 14, "w": 21, "uom": "cm" }, "status": "A" },  
    { "item": "sketch pad", "qty": 95, "size": { "h": 22.85, "w": 30.5, "uom": "cm" }, "status": "A" }  
])
```

## ✓ 특정 field 값을 업데이트 하기

```
inventory.update_one(  
  {"item": "canvas"},  
  {"$set": {"qty": 10}}  
)
```

## 결과 Document

```
{ item: "canvas", qty: 100, size: { h: 28, w: 35.5, uom: "cm" }, status: "A" }
```



```
{ item: "canvas", qty: 10, size: { h: 28, w: 35.5, uom: "cm" }, status: "A" }
```

## ✓ 특정 field를 제거하기

```
inventory.update_one(  
    {"item": "canvas"},  
    {"$unset": {"qty": True} }  
)
```

## 결과 Document

```
{ item: "canvas", qty: 100, size: { h: 28, w: 35.5, uom: "cm" }, status: "A" }
```



```
{ item: "canvas", size: { h: 28, w: 35.5, uom: "cm" }, status: "A" }
```

✓ 해당되는 document가 없다면 새로 추가하기

```
inventory.update_one(  
    { "item": "flash" },  
    { "$set": { "size": { "h": 10, "w": 8 }  
    , "status": "F" } },  
    True  
)
```

## 결과 Document

```
Null  
↓  
{ item: "flash", size: {h: 10, w: 8} ,status: "F"} }
```

✔ (참고) 이 외에도 사용할 수 있는 update 연산자

연산자 명	설명	형식
\$currentDate	<field>필드의 값으로 현재 timestamp나 date 값을 갖도록 추가한다.	\$currentDate: { <field>: <typeSpecification1>, ... }
\$inc	문서의 <field>필드의 값을 <증가시킬 값> 값만큼 증가시킨다.	\$inc: {<field>:<증가시킬 값>,...}
\$min	문서의 <field>필드의 값이 <최소값>이하면 <최소값>으로 설정한다.	\$min: {<field>:<최소값>,...}
\$max	문서의 <field>필드의 값이 <최대값>이상이면 <최대값>으로 설정한다.	\$min: {<field>:<최대값>,...}
\$mul	문서의 <field>필드의 값을 <배수>와 곱한 값으로 수정한다.	\$mul: {<field>:<배수>,...}
\$rename	문서의 <field>필드의 이름을 <새이름>으로 바꾼다	\$rename: {<field>:<새이름>,...}
\$setOnInsert	Upsert 값이 true 이면서 문서를 생성할 때만 사용되는 연산자. 쿼리와 \$set 연산자에 나온 필드와 값 이외의 필드와 값을 함께 설정한다.	\$setOnInsert: {<field>:<value>,...}

✔ (참고) 이 외에도 사용할 수 있는 배열 update 연산자

연산자 명	설명	예시
\$addToSet	배열안에 해당 값이 없다면 추가하고, 있다면 추가하지 않는다.	{ \$addToSet: { <field1>: <value1>, ... } }
\$pop	배열의 첫번째 혹은 마지막 요소를 삭제한다.	{ \$pop: { <field>: <-1   1>, ... } }
\$pull	쿼리에 해당하는 요소 하나를 제거한다.	{ \$pull: { <field1>: <value condition>, <field2>: <value condition>, ... } }
\$push	해당 요소를 배열에 추가한다.	{ \$push: { <field1>: <value1>, ... } }
\$pullAll	해당하는 값을 가지는 요소 전부를 제거한다.	{ \$pullAll: { <field1>: [ <value1>, <value2> ... ], ... } }
\$addToSet	배열안에 해당 값이 없다면 추가하고, 있다면 추가하지 않는다.	{ \$addToSet: { <field1>: <value1>, ... } }
\$pop	배열의 첫번째 혹은 마지막 요소를 삭제한다.	{ \$pop: { <field>: <-1   1>, ... } }

06

# 도큐먼트 삭제





## ✓ 하나의 문서를 찾아 삭제하기

### 명령어

```
result = collection.delete_one(  
    { query }  
)  
print(result.deleted_count)
```

### 결과

```
# 삭제된 문서 수  
1
```

**query로 검색하고 첫번째 문서를 삭제**

## ✓ 다수의 문서를 찾아 삭제하기

### 명령어

```
result = collection.delete_many(  
    { query }  
)  
print(result.deleted_count)
```

### 결과

```
# 삭제된 문서 수  
12
```

**query로 검색하고 다수의 문서를 삭제**

## ✔ 특정 field 값을 업데이트 하기

```
user.delete_one(  
  {"username": "karoid"}  
)
```

## 결과 Document

```
{username: "karoid", password: "1111"}
```



Null

## ✔ 특정 field 값을 업데이트 하기

```
user.delete_many(  
  {"password": "1111"}  
)
```

## 결과 Document

```
{username: "hello", password: "1111"},  
{username: "hi", password: "1111"}
```



Null

# 크레딧

/\* elice \*/

코스 매니저

이재성

콘텐츠 제작자

정승호

강사

정승호

감수자

정승호

디자이너

강혜정

# 연락처

TEL

070-4633-2015

WEB

<https://elice.io>

E-MAIL

[contact@elice.io](mailto:contact@elice.io)

